**Лабораторная работа 2. Комбинаторные алгоритмы решения оптимизационных задач**

**ЦЕЛЬ РАБОТЫ:** приобрести навыки разработки генераторов подмножеств, перестановок, сочетаний и размещений на С++; научиться применять разработанные генераторы для решения задач о рюкзаке (упрощенную, коммивояжера, об оптимальной загрузке судна и об оптимальной загрузке судна с центровкой.

**Ход работы**

## **Написание кода генератора подмножеств заданного множества.**

Код файла Combi.h приведен в листинге 1.1.

|  |
| --- |
| #pragma once  namespace combi  {  struct subset // генератор множества всех подмножеств  {  short n, // количество элементов исходного множества < 64  sn, // количество элементов текущего подмножества  \* sset; // массив индексов текущего подмножества  unsigned \_\_int64 mask; // битовая маска  subset(short n = 1); // конструктор(количество элементов исходного множества)  short getfirst(); // сформормировать массив индексов по битовой маске  short getnext(); // ++маска и сформировать массив индексов  short ntx(short i); // получить i-й элемент массива индексов  unsigned \_\_int64 count(); // вычислить общее количество подмножеств  void reset(); // сбросить генератор, начать сначала  };  } |

Листинг 1.1. Содержание файла Combi.h

Код файла Combi.cpp приведен в листинге 1.2.

|  |
| --- |
| #include "stdafx.h"  #include "Combi.h"  #include <algorithm>  namespace combi  {  subset::subset(short n)  {  this->n = n;  this->sset = new short[n];  this->reset();  };  void subset::reset()  {  this->sn = 0;  this->mask = 0;  };  short subset::getfirst()  {  \_\_int64 buf = this->mask;  this->sn = 0;  for (short i = 0; i < n; i++)  {  if (buf & 0x1) this->sset[this->sn++] = i;  buf >>= 1;  }  return this->sn;  };  short subset::getnext()  {  int rc = -1;  this->sn = 0;  if (++this->mask < this->count()) rc = getfirst();  return rc;  };  short subset::ntx(short i)  {  return this->sset[i];  };  unsigned \_\_int64 subset::count()  {  return (unsigned \_\_int64)(1 << this->n);  };  } |

Листинг 1.2. Содержание файла Combi.cpp

## **1.2. Написание кода генератора сочетаний.**

Код файла Combi.h приведен в листинге 1.3.

|  |
| --- |
| #pragma once  #include <algorithm>  #define NINF ((short)0x8000)  namespace combi  {  struct xcombination // генератор сочетаний (эвристика)  {  short n, // количество элементов исходного множества  m, // количество элементов в сочетаниях  \* sset; // массив индексов текущего сочетания  xcombination(  short n = 1, //количество элементов исходного множества  short m = 1 // количество элементов в сочетаниях  );  void reset(); // сбросить генератор, начать сначала  short getfirst(); // сформировать первый массив индексов  short getnext(); // сформировать следующий массив индексов  short ntx(short i); // получить i-й элемент массива индексов  unsigned \_\_int64 nc; // номер сочетания 0,..., count()-1  unsigned \_\_int64 count() const; // вычислить количество сочетаний  };  } |

Листинг 1.3. Содержание файла Combi.h

Код файла Combi.cpp приведен в листинге 1.4.

|  |
| --- |
| #include "stdafx.h"  #include "Combi.h"  #include <algorithm>  namespace combi  {  xcombination::xcombination(short n, short m)  {  this->n = n;  this->m = m;  this->sset = new short[m + 2];  this->reset();  }  void xcombination::reset() // сбросить генератор, начать сначала  {  this->nc = 0;  for (int i = 0; i < this->m; i++) this->sset[i] = i;  this->sset[m] = this->n;  this->sset[m + 1] = 0;  };  short xcombination::getfirst()  {  return (this->n >= this->m) ? this->m : -1;  };  short xcombination::getnext() // сформировать следующий массив индексов  {  short rc = getfirst();  if (rc > 0)  {  short j;  for (j = 0; this->sset[j] + 1 == this->sset[j + 1]; ++j)  this->sset[j] = j;  if (j >= this->m) rc = -1;  else {  this->sset[j]++;  this->nc++;  };  }  return rc;  };  short xcombination::ntx(short i)  {  return this->sset[i];  };  unsigned \_\_int64 fact(unsigned \_\_int64 x) { return (x == 0) ? 1 : (x \* fact(x - 1)); };  unsigned \_\_int64 xcombination::count() const  {  return (this->n >= this->m) ?  fact(this->n) / (fact(this->n - this->m) \* fact(this->m)) : 0;  }; |

Листинг 1.4. Содержание файла Combi.cpp

## **1.3. Написание кода генератора перестановок.**

Код файла Combi.h приведен в листинге 1.5.

|  |
| --- |
| #pragma once  #include <algorithm>  #define NINF ((short)0x8000)  namespace combi  {  struct permutation // генератор перестановок  {  const static bool L = true; // левая стрелка  const static bool R = false; // правая стрелка  short n, // количество элементов исходного множества  \* sset; // массив индексов текущей перестановки  bool\* dart; // массив стрелок (левых-L и правых-R)  permutation(short n = 1); // конструктор (количество элементов исходного множества)  void reset(); // сбросить генератор, начать сначала  \_\_int64 getfirst(); // сформировать первый массив индексов  \_\_int64 getnext(); // сформировать случайный массив индексов  short ntx(short i); // получить i-й элемент масива индексов  unsigned \_\_int64 np; // номер перествновки 0,... count()-1  unsigned \_\_int64 count() const; // вычислить общее кол. перестановок  };  } |

Листинг 1.5. Содержание файла Combi.h

Код файла Combi.cpp приведен в листинге 1.6.

|  |
| --- |
| #include "stdafx.h"  #include "Combi.h"  #include <algorithm>  namespace combi  {  permutation::permutation(short n)  {  this->n = n;  this->sset = new short[n];  this->dart = new bool[n];  this->reset();  };  void permutation::reset()  {  this->getfirst();  };  \_\_int64 permutation::getfirst()  {  this->np = 0;  for (int i = 0; i < this->n; i++)  {  this->sset[i] = i; this->dart[i] = L;  };  return (this->n > 0) ? this->np : -1;  };  \_\_int64 permutation::getnext() //  {  \_\_int64 rc = -1;  short maxm = NINF, idx = -1;  for (int i = 0; i < this->n; i++)  {  if (i > 0 &&  this->dart[i] == L &&  this->sset[i] > this->sset[i - 1] &&  maxm < this->sset[i]) maxm = this->sset[idx = i];  if (i < (this->n - 1) &&  this->dart[i] == R &&  this->sset[i] > this->sset[i + 1] &&  maxm < this->sset[i]) maxm = this->sset[idx = i];  };  if (idx >= 0)  {  std::swap(this->sset[idx],  this->sset[idx + (this->dart[idx] == L ? -1 : 1)]);  std::swap(this->dart[idx],  this->dart[idx + (this->dart[idx] == L ? -1 : 1)]);  for (int i = 0; i < this->n; i++)  if (this->sset[i] > maxm) this->dart[i] = !this->dart[i];  rc = ++this->np;  }  return rc;  };  short permutation::ntx(short i) { return this->sset[i]; };  unsigned \_\_int64 permutation::count() const { return fact(this->n); };  } |

Листинг 1.6. Содержание файла Combi.cpp

## **1.4. Написание кода генератора размещений.**

Код файла Combi.h приведен в листинге 1.7.

|  |
| --- |
| #pragma once  #include <algorithm>  #define NINF ((short)0x8000)  namespace combi  {  struct accomodation // генератор размещений  {  short n, // количество элементов исходного множества  m, // количество элементов в размещении  \* sset; // массив индесов текущего размещения  xcombination\* cgen; // указатель на генератор сочетаний  permutation\* pgen; // указатель на генератор перестановок  accomodation(short n = 1, short m = 1); // конструктор  void reset(); // сбросить генератор, начать сначала  short getfirst(); // сформировать первый массив индексов  short getnext(); // сформировать следующий массив индексов  short ntx(short i); // получить i-й элемент массива индексов  unsigned \_\_int64 na; // номер размещения 0, ..., count()-1  unsigned \_\_int64 count() const; // общее количество размещений  };  } |

Листинг 1.7. Содержание файла Combi.h

Код файла Combi.cpp приведен в листинге 1.8.

|  |
| --- |
| #include "stdafx.h"  #include "Combi.h"  #include <algorithm>  namespace combi  {  accomodation::accomodation(short n, short m)  {  this->n = n;  this->m = m;  this->cgen = new xcombination(n, m);  this->pgen = new permutation(m);  this->sset = new short[m];  this->reset();  }  void accomodation::reset()  {  this->na = 0;  this->cgen->reset();  this->pgen->reset();  this->cgen->getfirst();  };  short accomodation::getfirst()  {  short rc = (this->n >= this->m) ? this->m : -1;  if (rc > 0)  {  for (int i = 0; i <= this->m; i++)  this->sset[i] = this->cgen->sset[this->pgen->ntx(i)];  };  return rc;  };  short accomodation::getnext()  {  short rc;  this->na++;  if ((this->pgen->getnext()) > 0) rc = this->getfirst();  else if ((rc = this->cgen->getnext()) > 0)  {  this->pgen->reset(); rc = this->getfirst();  };  return rc;  };  short accomodation::ntx(short i)  {  return this->sset[i];  };  unsigned \_\_int64 accomodation::count() const  {  return (this->n >= this->m) ?  fact(this->n) / fact(this->n - this->m) : 0;  };  } |

Листинг 1.8. Содержание файла Combi.cpp

## **1.5. Решение задачи об оптимальной загрузке судна.**

Код файла Boat.h приведен в листинге 1.9.

|  |
| --- |
| #pragma once  #include "Combi.h"  int boat(  int V, // [in] максимальный вес груза  short m, // [in] количество мест для контейнеров  short n, // [in] всего контейнеров  const int v[], // [in] вес каждого контейнера  const int c[], // [in] доход от перевозки каждого контейнера  short r[] // [out] результат: индексы выбранных контейнеров  ); |

Листинг 1.9. Содержание файла Boat.h

Код файла Boat.cpp приведен в листинге 1.10.

|  |
| --- |
| #include "stdafx.h"  #include "Boat.h"  namespace boatfnc  {  int calcv(combi::xcombination s, const int v[]) // вес  {  int rc = 0;  for (int i = 0; i < s.m; i++) rc += v[s.ntx(i)];  return rc;  };  int calcc(combi::xcombination s, const int c[]) // доход  {  int rc = 0;  for (int i = 0; i < s.m; i++) rc += c[s.ntx(i)];  return rc;  };  void copycomb(short m, short\* r1, const short\* r2) // копировать  {  for (int i = 0; i < m; i++) r1[i] = r2[i];  };  }  int boat(  int V, // [in] максимальный вес груза  short m, // [in] количество мест для контейнеров  short n, // [in] всего контейнеров  const int v[], // [in] вес каждого контейнера  const int c[], // [in] доход от перевозки каждого контейнера  short r[] // [out] результат: индексы выбранных контейнеров  )  {  combi::xcombination xc(n, m);  int rc = 0, i = xc.getfirst(), cc = 0;  while (i > 0)  {  if (boatfnc::calcv(xc, v) <= V)  if ((cc = boatfnc::calcc(xc, c)) > rc)  {  rc = cc; boatfnc::copycomb(m, r, xc.sset);  }  i = xc.getnext();  };  return rc;  }; |

Листинг 1.10. Содержание файла Boat.cpp

Код файла Main.cpp приведен в листинге 1.11.

|  |
| --- |
| #include "stdafx.h"  #include <iostream>  #include <iomanip>  #include "Boat.h"  #include "Auxil.h"  #define NN 25  #define MM 5  int \_tmain(int argc, \_TCHAR\* argv[])  {  setlocale(LC\_ALL, "rus");  int v[NN], V = 1500, c[NN] = { 10, 15, 20, 25, 30, 25, 70, 30, 120, 100, 90, 50, 75,  150, 140, 20, 10, 130, 30, 10, 70, 120, 125, 30, 100};    auxil::start();  for (int i = 0; i < NN; i++)  {  v[i] = auxil::iget(100, 900);  }  short r[MM];  int cc = boat(  V, // [in] максимальный вес груза  MM, // [in] количество мест для контейнеров  NN, // [in] всего контейнеров  v, // [in] вес каждого контейнера  c, // [in] доход от перевозки каждого контейнера  r // [out] результат: индексы выбранных контейнеров  );  std::cout << std::endl << "- Задача о размещении контейнеров на судне";  std::cout << std::endl << "- общее количество контейнеров : " << NN;  std::cout << std::endl << "- количество мест для контейнеров : " << MM;  std::cout << std::endl << "- ограничение по суммарному весу : " << V;  std::cout << std::endl << "- вес контейнеров : ";  for (int i = 0; i < NN; i++) std::cout << std::setw(3) << v[i] << " ";  std::cout << std::endl << "- доход от перевозки : ";  for (int i = 0; i < NN; i++) std::cout << std::setw(3) << c[i] << " ";  std::cout << std::endl << "- выбраны контейнеры (0,1,...,m-1): ";  for (int i = 0; i < MM; i++) std::cout << r[i] << " ";  std::cout << std::endl << "- доход от перевозки : " << cc;  std::cout << std::endl << "- общий вес выбранных контейнеров : ";  int s = 0; for (int i = 0; i < MM; i++) s += v[r[i]]; std::cout << s;  std::cout << std::endl << std::endl;  system("pause");  return 0;  } |

Листинг 1.11. Содержание файла Main.cpp

Результат работы программы представлен на рисунке 1.1.

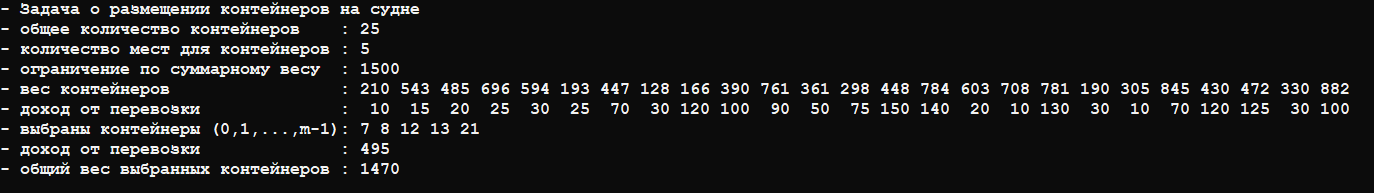


Рисунок 1.1. Результат работы программы

## **1.6. Исследование зависимости времени вычисления необходимого для решения задачи от размерности задачи.**

Код файла Main.cpp приведен в листинге 1.12.

|  |
| --- |
| #include "stdafx.h"  #include "Auxil.h"  #include <iostream>  #include <iomanip>  #include "Boat.h"  #include <time.h>  #define NN 35  #define MM 6  #define SPACE(n) std::setw(n)<<" "  int \_tmain(int argc, \_TCHAR\* argv[])  {  setlocale(LC\_ALL, "rus");  int v[NN], V = 1500;  int c[NN] = { 15, 26, 27, 43, 16, 26, 42, 22, 34, 12, 33, 30,  42,22, 34, 43, 16, 26, 14, 12, 25, 41, 17, 28,  12,45, 60, 41, 33, 11, 14, 12, 25, 41, 30};  auxil::start();  for (int i = 0; i < NN; i++)  {  v[i] = auxil::iget(100, 900);  }  short r[MM];  int maxcc = 0;  clock\_t t1, t2;  std::cout << std::endl << "-- Задача об оптимальной загрузке судна -- ";  std::cout << std::endl << "- ограничение по весу : " << V;  std::cout << std::endl << "- количество мест : " << MM;  std::cout << std::endl << "-- количество ------ продолжительность -- ";  std::cout << std::endl << " контейнеров вычисления ";  for (int i = 25; i <= NN; i++)  {  t1 = clock();  int maxcc = boat(V, MM, i, v, c, r);  t2 = clock();  std::cout << std::endl << SPACE(7) << std::setw(2) << i  << SPACE(15) << std::setw(5) << (t2 - t1);  }  std::cout << std::endl << std::endl;  system("pause");  return 0;  } |

Листинг 1.12. Содержание файла Main.cpp

Результат работы программы представлен на рисунке 1.2. Исследование зависимости представлено на рисунке 1.3.

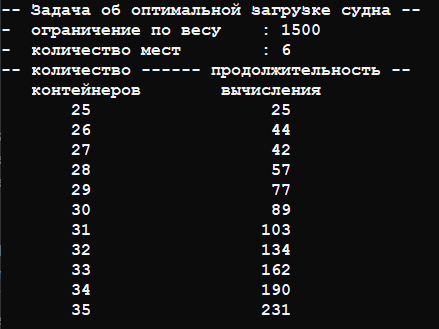


Рисунок 1.2. Результат работы программы

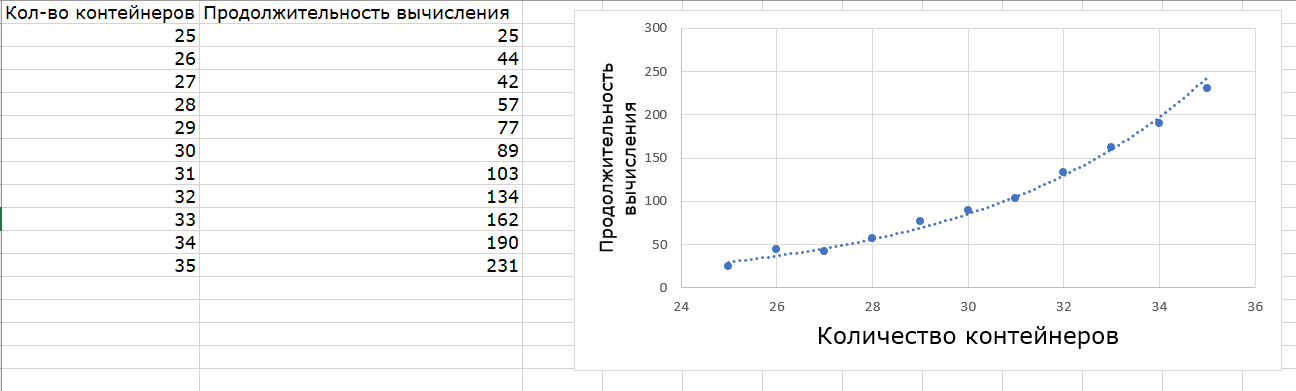


Рисунок 1.3 Исследование зависимости

Вывод: скорость выполнения программы нелинейно зависит от количества контейнеров.

## **1.7. Решение задачи о рюкзаке.**

Код файла Knapsack.h приведен в листинге 1.13.

|  |
| --- |
| // Knapsack.h  #pragma once  #include "Combi.h"  int knapsack\_s(  int V, // [in] вместимость рюкзака  short n, // [in] количество типов предметов  const int v[], // [in] размер предмета каждого типа  const int c[], // [in] стоимость предмета каждого типа  short m[] // [out] количество предметов  ); |

Листинг 1.13. Содержание файла Knapsack.h

Код файла Knapsack.cpp приведен в листинге 1.14.

|  |
| --- |
| #include "stdafx.h"  #include "Knapsack.h"  int calcv(combi::subset s, const int v[]) // объем в рюкзаке  {  int rc = 0;  for (int i = 0; i < s.sn; i++) rc += v[s.ntx(i)];  return rc;  };  int calcc(combi::subset s, const int v[], const int c[]) //стоимость в рюкзаке  {  int rc = 0;  for (int i = 0; i < s.sn; i++) rc += (v[s.ntx(i)] \* c[s.ntx(i)]);  return rc;  };  void setm(combi::subset s, short m[]) //отметить выбранные предметы  {  for (int i = 0; i < s.n; i++) m[i] = 0;  for (int i = 0; i < s.sn; i++) m[s.ntx(i)] = 1;  };  int knapsack\_s(  int V, // [in] вместимость рюкзака  short n, // [in] количество типов предметов  const int v[], // [in] размер предмета каждого типа  const int c[], // [in] стоимость предмета каждого типа  short m[] // [out] количество предметов каждого типа {0,1}  )  {  combi::subset s(n);  int maxc = NINF, cc = 0;  short ns = s.getfirst();  while (ns >= 0)  {  if (calcv(s, v) <= V)  if ((cc = calcc(s, v, c)) > maxc)  {  maxc = cc;  setm(s, m);  }  ns = s.getnext();  };  return maxc;  }; |

Листинг 1.14. Содержание файла Knapsack.cpp

Код файла Main.cpp приведен в листинге 1.15.

|  |
| --- |
| #include "stdafx.h"  #include <iostream>  #include "Combi.h"  #include "Knapsack.h"  #include "Auxil.h"  #define NN 18  int \_tmain(int argc, \_TCHAR\* argv[])  {  setlocale(LC\_ALL, "rus");  int V = 300, // вместимость рюкзака  v[NN], // размер предмета каждого типа  c[NN]; // стоимость предмета каждого типа  short m[NN]; // количество предметов каждого типа {0,1}  auxil::start();  for (int i = 0; i < NN; i++)  {  v[i] = auxil::iget(10, 300);  c[i] = auxil::iget(5, 55);  }  int maxcc = knapsack\_s(  V, // [in] вместимость рюкзака  NN, // [in] количество типов предметов  v, // [in] размер предмета каждого типа  c, // [in] стоимость предмета каждого типа  m // [out] количество предметов каждого типа  );  std::cout << std::endl << "-------- Задача о рюкзаке --------- ";  std::cout << std::endl << "- количество предметов : " << NN;  std::cout << std::endl << "- вместимость рюкзака : " << V;  std::cout << std::endl << "- размеры предметов : ";  for (int i = 0; i < NN; i++) std::cout << v[i] << " ";  std::cout << std::endl << "- стоимости предметов : ";  for (int i = 0; i < NN; i++) std::cout << v[i] \* c[i] << " ";  std::cout << std::endl << "- оптимальная стоимость рюкзака: " << maxcc;  std::cout << std::endl << "- вес рюкзака: ";  int s = 0; for (int i = 0; i < NN; i++) s += m[i] \* v[i];  std::cout << s;  std::cout << std::endl << "- выбраны предметы: ";  for (int i = 0; i < NN; i++) std::cout << " " << m[i];  std::cout << std::endl << std::endl;  system("pause");  return 0;  } |

Листинг 1.15. Содержание файла Main.cpp

Результат работы программы представлен на рисунке 1.4.

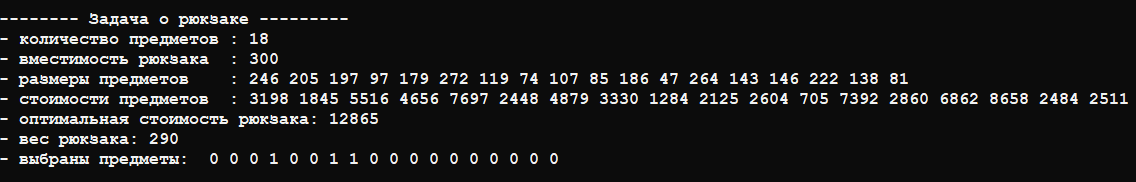


Рисунок 1.4. Результат работы программы

## **1.8. Исследование зависимости времени вычисления необходимого для решения задачи от размерности задачи.**

Код файла Main.cpp приведен в листинге 1.16.

|  |
| --- |
| #include "stdafx.h"  #include <iostream>  #include "Combi.h"  #include "Knapsack.h"  #include "Auxil.h"  #include <time.h>  #include <iomanip>  #define NN 20  int \_tmain(int argc, \_TCHAR\* argv[])  {  setlocale(LC\_ALL, "rus");  int V = 300, // вместимость рюкзака  v[NN], // размер предмета каждого типа  c[NN]; // стоимость предмета каждого типа  short m[NN]; // количество предметов каждого типа {0,1}  auxil::start();  for (int i = 0; i < NN; i++)  {  v[i] = auxil::iget(10, 300);  c[i] = auxil::iget(5, 55);  }  int maxcc = 0;  clock\_t t1, t2;  std::cout << std::endl << "-------- Задача о рюкзаке --------- ";  std::cout << std::endl << "- вместимость рюкзака : " << V;  std::cout << std::endl << "-- количество ------ продолжительность -- ";  std::cout << std::endl << " предметов вычисления ";  for (int i = 12; i <= NN; i++)  {  t1 = clock();  maxcc = knapsack\_s(V, i, v, c, m);  t2 = clock();  std::cout << std::endl << " " << std::setw(2) << i  << " " << std::setw(5) << (t2 - t1);  }  std::cout << std::endl << std::endl;  system("pause");  return 0;  } |

Листинг 1.16. Содержание файла Main.cpp

Результат работы программы представлен на рисунке 1.5. Исследование зависимости представлено на рисунке 1.6.

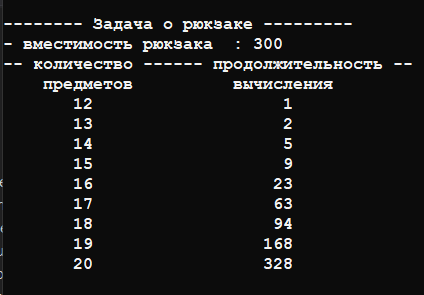


Рисунок 1.5. Результат работы программы

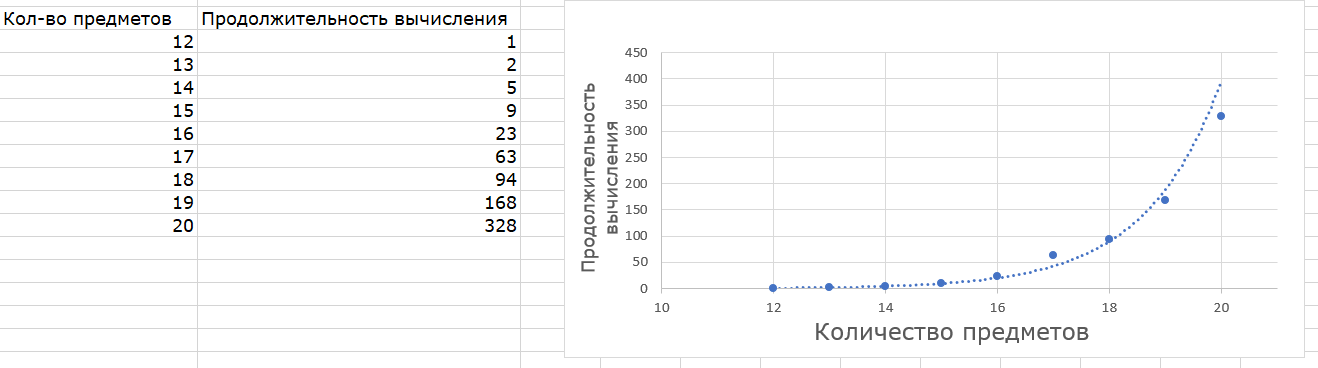


Рисунок 1.6 Исследование зависимости

Вывод: скорость выполнения программы нелинейно зависит от количества предметов.